Klausurersatz 17.01.2023
Informatik Q1
Francis Pennoyer

23.11.2022 / 25.11.2022

- Festlegen auf Bubblesort als zu programmierenden Sortieralgorithmus
- Allgemeines Recherchieren über den Bubblesort-Algorithmus
- Wiederholung Arrays
- Kurzes Auseinandersetzen mit dem Thema der Zeitkomplexität und deren Notation

Ergebnisse über den Bubblesort-Algorithmus:
https://de.wikipedia.org/wiki/Bubblesort (Erläuterung)
https://www.hackerearth.com/practice/algorithms/sorting/bubble-sort/visualize/ (Visualisierung)
https://www.youtube.com/watch?v=jMId4dq6p44 (Erklärung + Implemenatationsbeispiel)
https://www.youtube.com/watch?v=NKRO2GbjaAo&t=491s (Zeitkomplexität & Landau Notation)

30.11.2022 / 02.11.2022

- Einfinden in die gegebenen GUI-Vorlagen (MoD_sort.jfm etc.)
- Sammeln von Ideen, wie man den Bubblesort-Algorithmus in das Programm implementieren kann
    1. Den Index benutzen, um auf einzelne Array-Elemente zuzugreifen. Dafür die Indexwerte einer Variable zuweisen und eine Hauptschleife kreieren (s. Z. 225-238) Man kann sich diese Variablen als “Zeiger” vorstellen, der auf die benachbarten zu vergleichenden Werte zeigt.
    2. Nebenschleife sorgt für das Aufrücken des “Zeigers” – also dafür, dass beide Indexwerte um 1 erhöht werden (s. Sourcecode Z. 227)
    3. If-Statement prüft die Reihenfolge der Zahlenwerte auf das Sortierkriterium (s. Sourcecode Z. 230)

    3.2 Bei Erfüllung des Sortierkriteriums zurück in Hauptschleife und Indexe um erhöhen

    4. Bei Verletzung des Sortierkriteriums (also falscher Reihenfolge) Tausch einleiten (s. Sourcecode Z. 232-234). Zum Tauschen kleineren Wert in temporäre Variable “verstauen”, größere Wert auf Position des kleineren Wertes spiegeln – nun haben benachbarten Indexe den größeren Wert und der kleinere ist in der “tempVar” verstaut. Und schließlich Wert aus der temporären Variable an vorherige Stelle des größeren Wertes einsetzen (siehe im folgenden Beispiel oder im Video über Bubblesort-Implementierung (oben)).

[3, 1, 4, ...] (soll von klein nach groß geordnet sein)

^ ^

Variable j k

(j) = 3; (k) = 1; –> 1 in tempVar –> 3 in (k) –> (j) und (k) haben Wert 3,

woraus folgt: [3, 3, 4, ...]

–> tempVar in (j)

[1, 3, 4, ...]

^ ^

Variable j k –> j = 1; k = 3; –> [1, 3, 4, ...] – Tausch abgeschlossen

Zur besseren Veranschaulichung wurden im Beispiel zwei unterschiedliche Variablen verwendet. Im Sourcecode wird zum Beschreiben der benachbarten Werte der Index des Links liegenden Wertes j und der Index des Rechts liegenden Wertes j+1 verwendet. Im obigen Beispiel läge im "Ausgangs Array" [3, 1, 4, ...] der Zahlenwert 3 auf dem Index j = 0 und der Zahlenwert 1 auf dem Index j+1 = 1.

Quelle (abgesehen vom YouTube-Video vom Kanal “Programmieren Starten” über Bubblesort s. oben – 4. Link):

https://stackoverflow.com/questions/11644858/bubblesort-implementation

- Recherche über for-Schleifen und wie man diese verschachtelt sowie If-Statements

(Hilfsquellen: https://www.youtube.com/watch?v=jMId4dq6p44 und https://www.youtube.com/watch?v=gipOxNE6EMI)

03.12.2022 ***(außerhalb des Unterrichts)***

- Auftreten eines Problems:

Fehlendes UI-Element (NoClassDefFoundError: JNumberField).
Diese Klasse JNumberField ist weder in der UI-Vorlage gegeben noch in der Laufzeitumgebung (JRE) inkludiert.
Lösung: Im Internet gibt es eine Implementation der JNumberField Klasse von Christian Fries, die nun als ".java"-Datei im "src"-Ordner vorliegt und auf die das Programm zugreift.
[Link zur JNumberField-Klasse: https://www.finmath.net/finmath-lib/apidocs/net.finmath.lib/net/finmath/swing/JNumberField.html] gehe zu Datei (im GitHub Repo)

07.12.2022 / 09.12.2022 *(außerhalb des Unterrichts – Entfall)*

- Implementation der Anzahl der Vergleiche in das Programm

Ansatz 1: Getter und Setter für Länge des Arrays benutzen, um diese dann in Formel n*(n-1)/2 für n einzusetzen.
Beispiel: angenommen, das Array hat 10 verschiedene Werte: (10*(10-1)) /2 = 45
–> 45 Vergleiche
Ansatz 2: einen Zähler einbauen. Dafür eine Variable innerhalb des Konstruktors deklarieren und auf
"0" setzen. Für jeden Durchlauf der Schleife den "Zählerwert" um 1 erhöhen

(s. Sourcecode Z. 228).
Ansatz 2 einfacher umzusetzen und daher verwendet, um die Anzahl der Vergleiche ausdrucken zu lassen.

- Implementation der Anzahl an Verschiebungen, die nötig war, um die Zahlenliste zu sortieren

Gleiches Prinzip wie für die Anzahl der Vergleiche (Variable deklarieren und gleich "0" setzen, aber immer beim Durchlaufen der Schleife um 1 erhöhen – s. Sourcecode Z. 231.

14.12.22 / 16.12.22 *(außerhalb des Unterrichts – Entfall)*

Wichtig zu wissen: es wird ein zufälliges Double zwischen 0 und 1 ausgesucht, das dann mit 100 multipliziert wird. Dadurch entstehen die Array Werte zwischen 0 und 100

- Eliminieren von Duplikaten im Array

Ansatz 1: Spektrum an Zahlen vergrößern, von denen zufällige ausgewählt werden. Dadurch sinkt die Wahrscheinlichkeit für zwei gleiche Werte. Beispielsweise, statt 50 Zahlen von 0-100, 50 Zahlen von 0-1000 generieren. Dafür im Sourcecode in Zeile 186 *"100" mit "1000"* ersetzen. Dadurch würden Zahlen zwischen 0 und 1000 entstehen. Problem dabei ist, dass nicht nur weiterhin eine gewisse Wahrscheinlichkeit auf Duplikate bestehen bleibt, sondern auch noch die größeren Zahlen mehr Platz auf dem UI einnehmen und eventuell nicht mehr im Fenster stehen oder eine neue Zeile angelegt werden muss. Außerdem würden nicht alle 1001 möglichen Zahlen "ausgenutzt" werden, da ein Double mit 1000 multipliziert eine Zahl mit einer 0 als letzte Ziffer ergibt, sprich einer Vielfachen von 10 (Bsp.: 0,13*1000 = 130).

Ansatz 2: (s. Sourcecode Z. 188 f.) Nutzen von "Kandidaten" für Arraywerte (auch wieder mit zwei verschachtelten For-Schleifen): Voraussetzung: *"tbnoduplicates1 = true"*

- Für Index Variable deklarieren und gleich 0 setzen
- Der Variable des Indexes einen Kandidaten zuschreiben (irgendein Double zwischen 0 und 1, was mit 100 multipliziert wird und dann eine zufällige Zahl zwischen 1 und 100 ergibt)
- Schauen, ob der Kandidat bereits als ein Wert an einer vorherigen Stelle existiert
- **Wenn ja:** im Terminal folgendes Ausdrucken: "Generierte Zahl existiert bereits. Versuche erneut." und neuen Kandidaten generieren so lange, bis ein Kandidat gefunden ist, der im Array noch nicht existiert.
- **Wenn nein: if-Statement überspringen und** Kandidat in das Array bei jenem Index einsetzen und im Array eins aufrücken (Index um 1 erhöhen – wie auch in der Hauptschleife des Bubblesort-Algorithmus-Codes – durch *"Variable"++).*

Problem war, dass der ToggleButton "tbnoduplicates1" genau umgekehrt funktioniert. Daraus folgte, dass wenn die Variable des Typs Boolean "noDuplicates" = true gesetzt worden war, keine Duplikate entstehen sollten und dennoch kamen sie vor. In dem Fall wurde nämlich das if-Statement, das für die Eliminierung der Duplikate sorgt, übersprungen, obwohl der Button angeklickt war – also genau andersherum, wie eigentlich geplant. –> aus "noDuplicates = true" folgten Duplicates...
Ansatz 1: Als Lösung hätte man umständlicher Weise den Sourcecode nochmal großartig ändern können.
Ansatz 2: Stattdessen kann man im if-Statement (s. Z. 188) vor die Variable "noDuplicates" ein "!" setzen, um das Boolean zu invertieren. Dadurch ergibt sich folgendes:
Der Button "tbnoduplicates1" ist angeklickt, woraus folgt, dass das if-Statement übersprungen werden würde. Durch das Ausrufezeichen vor der Variable des ToggleButtons passiert genau das Gegenteil.

Auswertung der Hilfsmittel und Quellen

Wikipedia – Könnte für einen Einsteiger anfangs etwas schwierig zu verstehen sein, aufgrund der Fachsprache. In den ersten paar Zeilen wird direkt von einem “stabilen” Sortieralgorithmus gesprochen, der in-place arbeitet und eine Laufzeit von … hat.
Hackerearth.com – Ist zwar auf Englisch und daher womöglich auch für den einen oder anderen schwieriger zu verstehen, aber (abgesehen davon, dass man sich die Seite übersetzen lassen kann,) ist sie sehr informativ, zuverlässig und bietet außerdem eine wirklich gute Visualisierung des Bubblesort-Algorithmus. Diese Seite kann man weiterempfehlen, vor allem, weil sie neben Bubblesort auch noch einige andere Sortieralgorithmen erklärt.
YouTube-Videos vom Kanal Programmieren starten – In den Videos wird das jeweilige Thema ausführlich, aber gut verständlich erklärt und dargestellt, weshalb man ihm gut folgen kann.
Im Video, wo er den Bubblesort-Algorithmus in C# programmiert, könnte er dennoch etwas ausführlicher erklären, was genau welcher Teil des Codes bzw. Welcher Teil der Schleife genau macht. Einem Einsteiger, der sich noch nicht ganz mit for-Schleifen auskennt, könnte man daher noch ans Herz legen, sich mit for-Schleifen und deren Aufbau auseinanderzusetzen.
Das Video über die Landau-Notation ist ebenfalls sehr gut verständlich und informativ.

Stackoverflow.com – Abgesehen davon, dass es sich hierbei sowieso um eine bekannte und zuverlässige Quelle handelt, findet man wirklich zu sehr vielen Themen Code, den man schon fast einfach kopieren und einfügen kann. In diesem muss man aber etwas herunterscrollen, bis man die Antwort findet, die für diese Klausurersatzleistung als Orientierung verwendet wurde. Wenn man sich nicht mit seiner IDE oder der Programmiersprache auskennt, könnte man Probleme beim Implementieren bekommen.

Finmath.net – Relativ unübersichtlich, aber lieferte am Ende das benötigte Ergebnis, nämlich die Klasse JNumberField.

Github.com – Wird sozusagen als Speicher des Projektes benutzt. Man kann das Repository einfach als eine Zip-Datei herunterladen oder den Link verwenden, um dieses zu teilen.
Anhänge / GUI-Vorlagen – Relativ übersichtlich und gut nachvollziehbar.

17.01.2023 *(außerhalb des Unterrichts)*

Feinschliff des Projektes

- Ausführliches Kommentieren des Codes und dementsprechendes Anpassen der Zeilenangaben in der Dokumentation – nun gibt es auch Links, die direkt zu den Zeilen führen
- Hinzufügen eines Headers mit einer Vorlage aus dem Internet
- Readme erstellt
- IntelliJ für Formatierung des Codes (Versetzung bzw. Einrückung)
- Fertigstellen dieser Dokumentation sowie des Infomaterials

HIER DER LINK ZUM GITHUB REPO:

https://github.com/F-P-1611/sorting